

Universidade do Minho

Escola de Engenharia

Mestrado Integrado em Engenharia Informática

Unidade Curricular de Comunicações por Computador

Ano Letivo de 2022/2023

TP2 - Implementação de Sistema DNS

Grupo 3.08

A91775 José Pedro Batista Fonte

A94942 Miguel Velho Raposo

3 de janeiro de 2023

Resumo

O presente relatório é referente ao trabalho prático desenvolvido pelo grupo 08 do turno TP3 no âmbito da Unidade Curricular de Comunicações por Computador, lecionada no curso de Mestrado Integrado em Engenharia Informática no 1º Semestre do ano letivo 2022/2023.

O projeto visa a construção de um Sistema DNS e apresenta todas as etapas do seu desenvolvimento. O trabalho apresentado adiante foca-se nas , primeiramente, numa introdução aos conceitos abordados e o tema em estudo, assim como, objetivos, ferramentas de desenvolvimento. Depois aborda-se o sistema construído de modo a simular um sistema de DNS - isto inclui, todos os componentes, os seus requisitos, algumas decisões tomadas sobre o sistema, etc. Depois, apresenta-se as diferentes formatações que compõem os ficheiros do trabalho assim como os modelos de informação e comunicação seguidos. Para finalizar apresenta-se os diagramas de atividades que explicam o funcionamento dos componentes e um pequeno manual de utilização do sistema de modo a testar as suas funcionalidades.

Área de Aplicação: Sistema de DNS, Internet, Comunicações entre Computadores

Palavras-Chave: Domínios de DNS, Protocolos de Transportes (TCP/IP e UDP), Sockets, Multithreading, Locks, Queries, Servidor de Topo, Servidor Primário, Servidor Secundário, Servidor Resolução, Cliente.

Índice

Lista de Figuras	5
Lista de Tabelas	5
1 Introdução	6
1.1 Contextualização - A Internet	6
1.2 Tema em Estudo e Desenvolvimento - DNS	7
1.3 Objetivos	7
1.4 Ferramentas de Desenvolvimento do Software	9
2 Arquitetura do Sistema e Planeamento	10
2.1 Identificação dos Elementos do Sistema	10
2.2 Levantamento dos Requisitos do Sistema	11
2.2.1 Requisitos da 1ª Fase	11
2.2.2 Requisitos da 2ª Fase	12
2.3 Descrição da Topologia da Rede Concebida	13
2.4 Identificação dos Componentes de Software	17
3 Modelo de Informação	18
3.1 Especificação da Sintaxe e Semântica dos ficheiros	18
3.2 Estrutura das Mensagens de DNS	22
3.3 Tratamento de erros	24
3.4 Mecanismo de Codificação Binária	25
4 Modelo de Comunicação	27
4.1 Protocolos de Transporte	27
4.2 Processamento Iterativo	27
4.3 Processamento Recursivo	28
4.4 Processamento das Queries	29
4.5 Ambiente de Teste	29
4.5.1 Ficheiros de Configuração	30
4.5.2 Ficheiros de Base de Dados	31
5 Funcionamento dos Componentes	33
5.1 Execução Concorrente de Pedidos	33
5.1.1 Estado Partilhado e Zona de Exclusão Mútua	33
5.2 Processamento de uma querie	34
5.3 Cliente	34
5.4 Servidor Resolução (SR)	35
5.5 Servidor Secundário (SS)	35

5.6	Servidor Primário (SP)	36
5.7	Servidor de DNS Reverso	37
6	Manual de Utilização	38
7	Conclusão	39
8	Anexos	40

Lista de Figuras

1.1	Mapa parcial Internet	6
1.2	OSI & TCP/IP	6
2.1	Topologia da rede concebida	16
3.1	Lógica da mensagem DNS no sistema	23
4.1	Processamento Iterativo	28
4.2	Processamento Recursivo	29
5.1	Diagrama de Atividades do CL	34
5.2	Diagrama de Atividades do SR	35
5.3	Diagrama de Atividades do SS	36
5.4	Diagrama de Atividades do SP	37
8.1	Topologia da rede concebida	40
8.2	Legenda da rede concebida	40
8.3	Diagrama do Cliente	41
8.4	Diagrama do Servidor Resolução	42
8.5	Diagrama do Servidor Primário	43
8.6	Diagrama do Servidor Secundário	44

Lista de Tabelas

1 Introdução

O presente relatório é referente ao trabalho prático desenvolvido pelo grupo 8 do turno TP3 no âmbito da Unidade Curricular de Comunicações por Computador, lecionada no curso de Mestrado Integrado em Engenharia Informática no 1º Semestre do ano letivo 2022/2023.

O projeto visa o desenvolvimento de um Sistema DNS e apresenta todas as etapas do seu desenvolvimento. Sendo que este relatório é referente à primeira fase, o trabalho apresentado adiante foca-se, primeiramente, numa breve contextualização do tema e na especificação do caso em estudo, assim como os objetivos do grupo e as ferramentas de trabalho a utilizar.

1.1 Contextualização - A Internet

A Internet é um sistema global de redes de computadores interligados que utilizam um conjunto próprio de protocolos com o propósito de servir progressivamente usuários no mundo inteiro. O avanço exponencial da sociedade deve-se em grande parte à internet, que desbloqueou um novo mundo virtual, um mundo de conhecimento partilhado, de comunicação virtualmente instantânea, de entretenimento ilimitado e toda uma nova economia digital.

A Internet conta já com quase 5 bilhões de usuários (mais de 60% da população mundial) o que naturalmente eleva a exigência nos protocolos que a regem. A stack de protocolos TCP/IP é o conjunto de protocolos de comunicação que regem a internet.

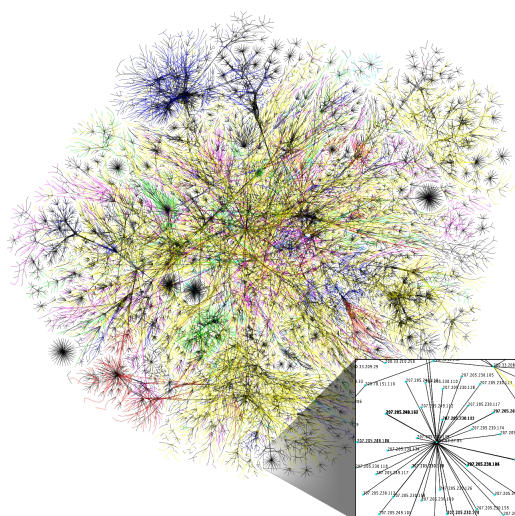


Figura 1.1: Mapa parcial Internet

OSI seven-layer model		TCP/IP four-layer model
Application	+	Application
Presentation		
Session		
Transport	+	Transport
Network	+	Internet
Data-link	+	Network
Physical		

Figura 1.2: OSI & TCP/IP

1.2 Tema em Estudo e Desenvolvimento - DNS

O tema proposto pela equipa docente foi a implementação de um sistema DNS, semelhante ao mundialmente implementado. O Sistema de Nomes de Domínio, mais conhecido pela nomenclatura em Inglês Domain Name System (DNS), é um sistema hierárquico e distribuído de gestão de nomes para computadores, serviços ou qualquer máquina conectada à Internet ou a uma rede privada. O DNS na sua utilização mais convencional associa nomes de domínios, mais facilmente memorizáveis, a endereços IP numéricos, necessários à localização e identificação de serviços e dispositivos.

Na prática o DNS funciona da seguinte forma: por exemplo, se o José quiser aceder à página do Departamento de Informática pesquisa `www.di.uminho.pt` e não pelo seu endereço IP `193.136.19.38`.

A pesquisa por nomes torna possível o que seria, de outro modo, uma pesquisa impossível. O ser humano consegue, em média, memorizar 5 a 9 números seguidos a curto prazo. Sendo que o IPV4 pode ter até 12 números, a tarefa de saber mais do 10 endereços torna-se extremamente difícil.

O sistema DNS permite ao ser humano ter apenas conhecimento do nome do que pretende pesquisar e com uma estrutura de domínios e subdomínios é possível mapear o endereço IP do servidor onde está alojado o serviço pretendido.

1.3 Objetivos

Objetivos Gerais

- Consolidação dos conhecimentos sobre o serviço DNS da arquitetura TCP/IP e sobre os protocolos de transporte UDP e TCP.
- Saber especificar um PDU e as primitivas de serviço dum protocolo aplicacional utilizando um protocolo de transporte orientado à conexão e um protocolo de transporte não orientado à conexão.
- Consolidar competências de programação de aplicações distribuídas utilizando o paradigma dos sockets.

Objetivos para a 1ª Fase - Arquitetura do Sistema e Planeamento

- **Arquitetura do Sistema:** Descrição do sistema e especificação completa dos requisitos funcionais esperados para cada elemento do sistema; descrição dos componentes de software a utilizar e os módulos que os compõem.
- **Modelo de Informação:** Especificação completa da sintaxe e da semântica de todos os ficheiros e do comportamento dos elementos em situação de erro de leitura, do PDU/mensagens DNS e eventual mecanismo de codificação binária.
- **Modelo de Comunicação:** Especificação completa de todas as interações possíveis e do comportamento dos elementos em situação de erro.
- **Planeamento do Ambiente de Teste:** Descrição e especificação completa do ambiente de testes a utilizar na fase 2, incluindo o conteúdo dos ficheiros de configuração e de dados.
- **Protótipo SP e SS em modo debug:** Implementação dum componente que implemente um SP e um SS em modo debug conciso que suporte a receção de queries e resposta, no mínimo, com a inclusão simples de apenas o campo de resposta direta; inclusão do mecanismo de transferência de zona simples sem verificação das versões das bases de dados.
- **Protótipo CL em modo debug:** Implementação dum componente que implemente um CL em modo debug conciso que suporte o envio de queries e permita, no mínimo, visualizar o campo de resposta direta.

Objetivos para a 2ª Fase - Implementação e Testes

- **Atualização da Arquitetura do Sistema:** Correção e adição de novos aspetos na especificação da arquitetura.
- **Atualização do Modelo de Informação:** Correção e adição de novos aspetos na especificação respetiva.
- **Atualização do Modelo de Comunicação:** Correção e adição de novos aspetos na especificação respetiva.
- **Planeamento do Ambiente de Teste:** Correção e adição de novos aspetos da descrição e especificação completa do ambiente de testes.
- **Implementação de SP, SS, ST, SDT e SR em modo debug e normal:** Implementação completa do componente que implemente um SP, tendo em consideração que um ST, um SDT e um SR são apenas casos particulares de implementações de SP ou SS configurados de forma adequada e incluindo um mecanismo de cache.

- **CL em modo debug e modo normal** : Implementação completa dum componente que implemente um CL .
- **Implementação do Ambiente de Teste** : Implementação e teste da especificação construída 1ª Fase .

1.4 Ferramentas de Desenvolvimento do Software

Para assegurar as funcionalidades pretendidas, o grupo utilizou as seguintes ferramentas para construir,:

- Modelação, Concepção e Análise da Rede: Core, Wireshark.
- Software de Desenvolvimento: Python, Visual Studio Code.
- Software de Gestão do Projeto: Overleaf, GitHub.

2 Arquitetura do Sistema e Planeamento

Neste capítulo descreve-se em detalhe o sistema desenvolvido. Primeiramente, descreve-se os elementos que compõe o sistema e todos os requisitos que devem cumprir. De seguida, expõe-se a topologia concebida para responder aos requisitos e algumas decisões tomadas. Por último, descreve-se os componentes de software, a sua organização e função.

2.1 Identificação dos Elementos do Sistema

Tendo em consideração as normas que especificam o sistema DNS da Internet, podemos identificar quatro tipos de elementos fundamentais que podem interagir: Servidor Primário (SP), Servidor Secundário (SS), Servidor de Resolução (SR) e Cliente (CL). Outros elementos fundamentais são os Servidores de Topo(ST) e os Servidores de Domínio de Topo(SDT).

- **Cliente (CL)** : Uma aplicação cliente de DNS é o processo que precisa da informação da base de dados de DNS dum determinado domínio. Envia queries de DNS e recebe respostas.
- **Servidor de Resolução(SR)** : Servidor DNS que responde a, e efetua, queries DNS sobre qualquer domínio, mas que não tem autoridade sobre nenhum pois serve apenas de intermediário. Normalmente guarda em cache as respostas para agilizar o resolução de DNS.
- **Servidor Secundário (SS)** : Servidor DNS que responde a, e efetua, queries DNS além de ter autorização e autoridade para possuir (e tentar manter atualizada) uma réplica da base de dados original do SP autoritativo dum domínio DNS
- **Servidor Primário (SP)** : Servidor DNS que responde a, e efetua, queries DNS e que tem acesso direto à base de dados dum domínio DNS, sendo a autoridade que o gere.
- **Servidor de Domínio de Topo(SDT)** : Servidor autoritativo sobre um domínio, isto é, responde tem autorização para responder a queries sobre esse domínio. No contexto do trabalho os SPs e SSs autoritativos sobre os domínios são SDTs.
- **Servidor de Topo(ST)** : Servidor de root de DNS, que redireciona para os SDTs. Com um funcionamento semelhante a um SP mas tem na sua base de dados os endereços de IP dos SDTs.

- **Servidor Reverse** : Servidor reverse de DNS, que através de um IP determina o nome do domínio associada, isto é, o oposto da pesquisa DNS.

2.2 Levantamento dos Requisitos do Sistema

De modo a implementar um sistema funcional é necessário primeiramente se faça um levantamento de requisitos do sistema. Os requisitos de cada fase alinham-se com os objetivos do enunciado.

2.2.1 Requisitos da 1ª Fase

- **Requisitos Gerais**

- Cliente, Servidor Primário e Servidor Secundário com comunicação funcional
- Transferência de Zona entre SP e SS
- Registo da atividade dos Servidores em ficheiros de logs
- Criar cache, criar funções da cache (procurar, registar, eliminar), implementar a leitura do ficheiro DB para a cache

- **Cliente (CL)**

- **Input** : {endereço IP} {nome} {tipo valor} {(R)}
- Ler o input
- Construi queries de DNS válidas
- Enviar Query
- Receber uma resposta da Query
- **Output** : Resposta à query

- **Servidor Secundário (SS)** :

- **Input** : Linha de comando. {ficheiro de configuração} {porta de atendimento} {timeout}
- Ler o input
- Ler ficheiro de configuração
- Copiar a DB do SP do seu domínio

- Criar e guardar em cache a BD copiada
- Receber queries
- Processar as queries
- Enviar resposta
- **Servidor Primário (SP) :**
 - **Input** : {ficheiro de configuração} {porta de atendimento} {timeout}
 - Ler o input
 - Ler ficheiro de configuração
 - Criar cache e ler a sua base de dados
 - Enviar uma cópia da DB para o SS
 - Receber queries
 - Processar as queries
 - Enviar as respostas

2.2.2 Requisitos da 2ª Fase

- **Requisitos Gerais**
 - Processamento concorrente de pedidos nos servidores (multithreading)
 - Implementar locks nos sockets, nos ficheiros e nas estruturas de dados
 - Criar um ReadWrite lock para melhorar eficiência.
 - Implementar o *Reverse Mapping*
 - Processamento Iterativo e Recursivo
 - Escrever em Logs
- **Servidor de Topo (ST)**
 - **Input** : {ficheiro de configuração} {porta de atendimento} {timeout}
 - Processar querie sobre a BD dos STs. Envia informação sobre o SdT do domínio.
- **Servidor de domínio de topo (SDT)**
 - **Input** : {ficheiro de configuração} {porta de atendimento} {timeout}

- Processar queries sobre a BD dos domínios de Topo.
 - * Se a query for sobre o domínio responde com as informações que tem.
 - * Se a query não for sobre o domínio mas for sobre o seu subdomínio, deve responder com as informações do SP do seu subdomínio.
- **Servidor de Resolução (SR)**
 - **Input** : {ficheiro de configuração } {porta de atendimento} {timeout}
 - Registar toda atividade nos ficheiros logs
 - Ler o input
 - Ler ficheiro de configuração
 - Recebe a query do cliente e responde
 - Controla o fluxo de queries no processo iterativo
 - Cache própria que agiliza a resposta a queries
- **Servidor de Reverse (SR)**
 - **Input** : {ficheiro de configuração } {porta de atendimento} {timeout}
 - Criar Base de Dados do Reverse
 - Receber/Processar/Enviar queries
 - Implementar no processo iterativo

2.3 Descrição da Topologia da Rede Concebida

A Topologia de Rede criada deve seguir algumas normas impostas no enunciado do trabalho. Os requisitos são os seguintes:

- Dois Servidores de Topo;
- Servidores Domínio de Topo para dois domínios de topo;
- Um domínio de topo nomeado de .reverse onde estarão pendurados os domínios de DNS reverso;
- Um subdomínio em cada domínio de topo;
- Um Servidor Primário e dois Sevidores Secundários paraara cada domínio de topo e os seus subdomínios

- Num dos subdomínios devem instalar um SR que, pelo menos, implemente caching positivo;
- Numa host qualquer instalar o programa que implementa o CL;
- Para um dos domínios de topo devem instalar-se e configurar-se os respetivos servidores SP e SS para implementar o respetivo reverse mapping;

O backbone da topologia criada é retirada da topologia disponibilizada no TP1, no entanto corrigiu-se os problemas de conexão. Assim sendo, o core da topologia apresenta uma rede de 7 routers interligados, que representam uma distância física entre servidores.

De acordo com os requisitos é necessário 2 domínios de topo e 1 subdomínio em cada um. O grupo optou pelos seguintes domínios: **bra.** e **brg.**, e subdomínios: **bcl.bra.** e **mdd.brg.** - inspirados na localização dos autores do trabalho, o José Fonte oriundo de Barcelos (bcl.), Braga(bra.) e o Miguel Raposo de Miranda do Douro (mdd.), Bragança(brg.).

O sistema implementa obrigatoriamente dois servidores de topo ligados diretamente ao core, e um servidor de resolução chamado **.reverse**. Cada domínio e subdomínio é composto por um Servidor Primário(SP) e por 2 Servidores Secundários(SS). Os SPs também desempenham função de SDT e ST dependendo do domínio onde se encontram. O grupo também inclui dois servidores de Mail (M) e um de Web(W) por cada domínio e subdomínio. Deste modo a topologia apresenta os seguintes servidores:

- **Servidores de Topo** (rosa)

- ST1 : 10.0.13.10
- ST2 : 10.0.14.10

- **Servidores de Reverse** (cinzento)

- **.reverse** : 10.0.12.12

- **Domínio bra.** (vermelho)

- SP-bra : 10.0.12.10
- SDT-bra : 10.0.12.10
- SS1-bra : 10.0.17.12
- SS2-bra : 10.0.16.11
- M1-bra : 10.0.19.10
- M2-bra : 10.0.16.12
- W-bra : 10.0.18.11

- **Subdomínio bcl.brg.** (azul)

- SP-bcl : 10.0.17.10
- SDT-bcl : 10.0.17.10
- SS1-bcl : 10.0.20.11
- SS2-bcl : 10.0.21.12
- M1-bcl : 10.0.20.12
- M2-bcl : 10.0.21.11
- W-bcl : 10.0.19.12

- **Domínio** brg. (verde)

- SP-brg : 10.0.20.10
- SDT-brg : 10.0.20.10
- SS1-brg : 10.0.15.11
- SS2-brg : 10.0.12.11
- M1-brg : 10.0.15.12
- M2-brg : 10.0.18.12
- W-brg : 10.0.15.10

- **Subdomínio** mdd.brg. (amarelo)

- SP-mdd : 10.0.21.10
- SDT-mdd : 10.0.21.10
- SS1-mdd : 10.0.18.10
- SS2-mdd : 10.0.17.11
- SR-mdd : 10.0.23.10
- M1-mdd : 10.0.18.13
- M2-mdd : 10.0.19.11
- W-mdd : 10.0.15.10

Quanto aos domínios e subdomínios decidiu-se separá-los por vários switches que estão ligados ao core, garantindo que nenhum SP e SS do mesmo domínio/subdomínio estão ligados ao mesmo switch, o mesmo se aplica aos servidores mail do mesmo domínio/subdomínio.

Esta decisão garante que caso haja erros nas ligações ou em componentes da rede existe sempre um backup localizado noutro local físico, para assim, manter estabilidade e fiabilidade na rede.

Quanto à localização do servidor resolução criamos apenas um servidor de resolução (dentro do subdomínio mdd.brg.).

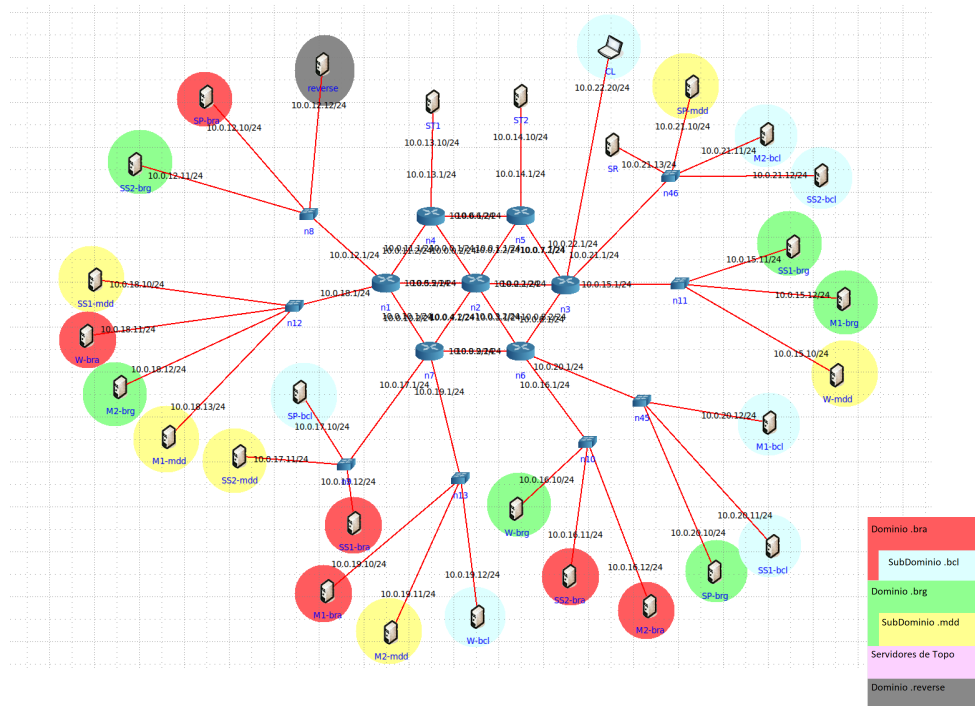


Figura 2.1: Topologia da rede concebida

2.4 Identificação dos Componentes de Software

A estrutura de software construída é a seguinte:

- **Ficheiros dos Componentes Principais:**

- `client.py` : executa o cliente
- `server.py` : inicializa o servidor correto de acordo com o ficheiro de configuração
- `sr.py` : inicializa o servidor de resolução com as suas configurações
- `sp.py` : inicializa o servidor primário com as suas configurações
- `ss.py` : inicializa o servidor secundário com as suas configurações

- **Ficheiros de Estruturas complementares:**

- `logs.py` : estrutura de dados dos logs e métodos de escrita para os ficheiros(e em modo DEBUG)
- `cache.py` : estrutura de dados da cache e métodos auxiliares da cache(procurar, inserir, limpar)
- `query.py` : estrutura de dados de uma query, métodos de parse e formatação de queries.
- `config.py` : todos os parsers: parser do ficheiro de configuração, do input do Cliente.
- `rwlock.py` : ficheiro do ReadWrite lock.

3 Modelo de Informação

O Modelo de Informação trata da sintaxe e semântica de todos os ficheiros configuração do sistema, das mensagens de DNS enviadas e do seu mecanismo de codificação binária.

3.1 Especificação da Sintaxe e Semântica dos ficheiros

Para este projeto são definidos alguns ficheiros de configuração, de dados e de log com uma sintaxe predefinida. Os ficheiros de configuração são apenas lidos e processados no arranque do componente de software a que dizem respeito e moldam o seu comportamento. Os ficheiros de dados também são consultados apenas no arranque e a sua informação é armazenada em memória. Os ficheiros de log mantêm um registo de atividade de todos os componentes.

Se for necessário atualizar o comportamento dos servidores com informação modificada nos ficheiros de configuração ou de dados que lhes dizem respeito a única alternativa é reiniciar esses servidores.

Ficheiros de Configuração dos Servidores SP, SS e SR

Este ficheiro tem uma sintaxe com as seguintes regras:

- As linhas começadas por '#' são consideradas comentários e são ignoradas;
- As linhas em branco também devem ser ignoradas;
- Deve existir uma definição de parâmetro de configuração por cada linha seguindo esta sintaxe:

{parâmetro} {tipo do valor} {valor associado ao parâmetro}

Tipos de valores aceites (todas as referências a domínios, quer nos parâmetros quer nos valores, são considerado nomes completos):

- **DB** – o valor indica o ficheiro da base de dados com a informação do domínio indicado no parâmetro (o servidor assume o papel de SP para este domínio);
- **SP** – o valor indica o endereço IP[:porta] do SP do domínio indicado no parâmetro (o

servidor assume o papel de SS para este domínio);

- **SS** – o valor indica o endereço IP[:porta] dum SS do domínio indicado no parâmetro (o servidor assume o papel de SP para este domínio) e que passa a ter autorização para pedir a transmissão da informação da base de dados (transferência de zona); podem existir várias entradas para o mesmo parâmetro (uma por cada SS do domínio);
- **DD** – o valor indica o endereço IP[:porta] dum SR, dum SS ou dum SP do domínio por defeito indicado no parâmetro; quando os servidores que assumem o papel de SR usam este parâmetro é para indicar quais os domínios para os quais devem contactar directamente os servidores indicados se receberem queries sobre estes domínios (quando a resposta não está em cache), em vez de contactarem um dos ST; podem existir várias entradas para o mesmo parâmetro (uma por cada servidor do domínio por defeito); quando os servidores que assumem o papel de SP ou SS usam este parâmetro é para indicar os únicos domínios para os quais respondem (quer a resposta esta em cache ou não), i.e., nestes casos, o parâmetro serve para restringir o funcionamento dos SP ou SS a responderem apenas a queries sobre os domínios indicados neste parâmetro;
- **ST** - o valor indica o ficheiro com a lista dos ST (o parâmetro deve ser igual a “root”);
- **LG** - o valor indica o ficheiro de log que o servidor deve utilizar para registar a atividade do servidor associada ao domínio indicado no parâmetro; só podem ser indicados domínios para o qual o servidor é SP ou SS; tem de existir pelo menos uma entrada a referir um ficheiro de log para toda a atividade que não seja directamente referente aos domínios especificados noutras entradas LG (neste caso o parâmetro deve ser igual a “all”).

Ficheiro com a Lista de Servidores de Topo

Este ficheiro tem a lista de Servidores de Topo e a sua predefinição indica que em cada linha deve existir um endereço IP[:porta] de um ST.

As linhas começadas por ‘#’ ou em branco devem ser ignoradas.

Ficheiros de log

Estes ficheiros registam toda a atividade relevante do componente; deve existir uma entrada de log por cada linha do ficheiro. A sintaxe de cada entrada é a seguinte :

{etiqueta temporal} {tipo de entrada} {endereço IP[:porta]} {dados da entrada}

A etiqueta temporal é a data e hora completa do sistema operativo na altura em que aconteceu a atividade registada e não a data e hora em que foi registada. Os tipos de entradas aceites são:

- **QR/QE** – foi recebida/enviada uma query do/para o endereço indicado; os dados da entrada devem ser os dados relevantes incluídos na query; a sintaxe dos dados de entrada é a mesma que é usada no PDU de query no modo debug de comunicação entre os elementos;
- **RP/RR** – foi enviada/recebida uma resposta a uma query para o/do endereço indicado; os dados da entrada devem ser os dados relevantes incluídos na resposta à query; a sintaxe dos dados de entrada é a mesma que é usada no PDU de resposta às queries no modo debug de comunicação entre os elementos;
- **ZT** – foi iniciado e concluído corretamente um processo de transferência de zona; o endereço deve indicar o servidor na outra ponta da transferência; os dados da entrada devem indicar qual o papel do servidor local na transferência (SP ou SS) e, opcionalmente, a duração em milissegundos da transferência e o total de bytes transferidos;
- **EV** – foi detetado um evento/atividade interna no componente; o endereço deve indicar 127.0.0.1 (ou localhost ou @); os dados da entrada devem incluir informação adicional sobre a atividade reportada (por exemplo, ficheiro de configuração/dados/ST lido, criado ficheiro de log, etc.);
- **ER** – foi recebido um PDU do endereço indicado que não foi possível decodificar corretamente; opcionalmente, os dados da entrada podem ser usados para indicar informação adicional (como, por exemplo, o que foi possível decodificar corretamente e em que parte/byte aconteceu o erro);
- **EZ** – foi detetado um erro num processo de transferência de zona que não foi concluída corretamente; o endereço deve indicar o servidor na outra ponta da transferência; os dados da entrada devem indicar qual o papel do servidor local na transferência (SP ou SS);
- **FL** – foi detetado um erro no funcionamento interno do componente; o endereço deve indicar 127.0.0.1; os dados da entrada devem incluir informação adicional sobre a situação de erro (por exemplo, um erro na decodificação ou incoerência dos parâmetros de algum ficheiro de configuração ou de base de dados);
- **TO** – foi detetado um timeout na interação com o servidor no endereço indicado; os dados da entrada devem especificar que tipo de timeout ocorreu (resposta a uma query ou tentativa de contato com um SP para saber informações sobre a versão da base de dados ou para iniciar uma transferência de zona);
- **SP** – a execução do componente foi parada; o endereço deve indicar 127.0.0.1; os dados da entrada devem incluir informação adicional sobre a razão da paragem se for possível obtê-la;
- **ST** – a execução do componente foi iniciada; o endereço deve indicar 127.0.0.1; os dados da entrada devem incluir informação sobre a porta de atendimento, sobre o valor do timeout usado (em milissegundos) e sobre o modo de funcionamento (modo “shy” ou modo debug).

Ficheiros de Dados do Servidor Primário

Este ficheiro tem uma sintaxe com as seguintes regras:

- As linhas começadas por '#' são consideradas comentários e são ignoradas;
- As linhas em branco também devem ser ignoradas;
- Deve existir uma definição de parâmetro de dados por cada linha seguindo esta sintaxe:

{parâmetro} {tipo do valor} {valor} {tempo de validade} {prioridade}

O *tempo de validade* (TTL) é o tempo máximo em segundos que os dados podem existir numa cache dum servidor (tanto serve para cache normal como para cache negativa, se for suportada). Quando o TTL não é suportado num determinado tipo, o seu valor deve ser igual a zero.

O campo *prioridade* é um valor inteiro menor que 256 e que define uma ordem de prioridade de vários valores associados ao mesmo parâmetro. Quanto menor o valor, maior a prioridade. Para parâmetros com um único valor ou para parâmetros em que todos os valores têm a mesma prioridade, o campo não deve existir.

Os nomes completos de e-mail, domínios, servidores e hosts devem terminar com um '.' (exemplo de nome completo de domínio: example.com.). Quando os nomes não terminam com '.' subentende-se que são concatenados com um prefixo por defeito definido através do parâmetro @ do tipo DEFAULT.

Tipos de valores a suportar (os tipos marcados com '*' são de implementação opcional e os tipos de valores que devem suportar o campo da prioridade são indicados explicitamente):

- **DEFAULT*** – define um nome (ou um conjunto de um ou mais símbolos) como uma macro que deve ser substituída pelo valor literal associado (não pode conter espaços nem o valor dum qualquer parâmetro DEFAULT); o parâmetro @ é reservado para identificar um prefixo por defeito que é acrescentado sempre que um nome não apareça na forma completa (i.e., terminado com '.'); o valor de TTL deve ser zero;
- **SOASP** – o valor indica o nome completo do SP do domínio (ou zona) indicado no parâmetro;
- **SOADMIN** – o valor indica o endereço de e-mail completo do administrador do domínio (ou zona) indicado no parâmetro; o símbolo '@' deve ser substituído por '.' e '.' no lado esquerdo de '@' devem ser antecidos de '\';
- **SOASERIAL** – o valor indica o número de série da base de dados do SP do domínio (ou zona) indicado no parâmetro; sempre que a base de dados é alterada este número deve ser incrementado;

- **SOAREFRESH** – o valor indica o intervalo temporal em segundos para um SS perguntar ao SP do domínio indicado no parâmetro qual o número de série da base de dados dessa zona;
- **SOARETRY** – o valor indica o intervalo temporal para um SS voltar a perguntar ao SP do domínio indicado no parâmetro qual o número de série da base de dados dessa zona, após um timeout;
- **SOAEXPIRE** – o valor indica o intervalo temporal para um SS deixar de considerar a sua réplica da base de dados da zona indicada no parâmetro como válida, deixando de responder a perguntas sobre a zona em causa, mesmo que continue a tentar contactar o SP respetivo;
- **SOAEXPIRE** – o valor indica o intervalo temporal para um SS deixar de considerar a sua réplica da base de dados da zona indicada no parâmetro como válida, deixando de responder a perguntas sobre a zona em causa, mesmo que continue a tentar contactar o SP respetivo;
- **NS** – o valor indica o nome dum servidor que é autoritativo para o domínio indicado no parâmetro, ou seja, o nome do SP ou dum dos SS do domínio; este tipo de parâmetro suporta prioridades;
- **A** – o valor indica o endereço IPv4 dum host/servidor indicado no parâmetro como nome; este tipo de parâmetro suporta prioridades;
- **CNAME** – o valor indica um nome canónico (ou alias) associado ao nome indicado no parâmetro; um nome canónico não deve apontar para um outro nome canónico nem podem existir outros parâmetros com o mesmo valor do nome canónico;
- **MX** – o valor indica o nome dum servidor de e-mail para o domínio indicado no parâmetro; este tipo de parâmetro suporta prioridades;
- **PTR** – o valor indica o nome dum servidor/host que usa o endereço IPv4 indicado no parâmetro; a indicação do IPv4 é feita como nos domínios de DNS reverso (rDNS) quando se implementa reverse-mapping;

3.2 Estrutura das Mensagens de DNS

Todas as interações assíncronas (não orientadas à conexão) possíveis neste sistema são feitas através de mensagens DNS encapsuladas no protocolo UDP. Todas as mensagens DNS devem ser implementadas usando a sintaxe da mesma unidade de dados protocolar (PDU). Uma mensagem DNS deve ter um cabeçalho de tamanho fixo e uma parte de dados que deve ocupar até 1 KByte. A parte de dados contém sempre quatro partes distintas: i) os dados duma query original; ii) os resultados diretos a essa query original; iii) informação sobre os servidores que têm informação autoritativa sobre os dados da resposta e iv) informação adicional indiretamente ligada aos resultados ou aos dados da query original.

Ver a Figura 3.1 com a representação lógica duma mensagem DNS usada neste sistema.

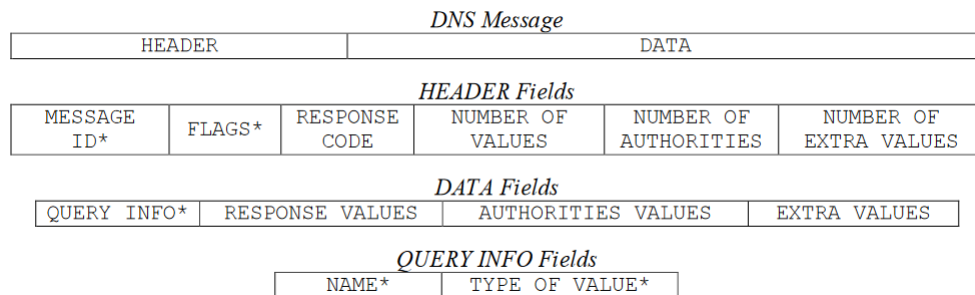


Figura 3.1: Lógica da mensagem DNS no sistema

Os campos do cabeçalho devem ser implementados de tal forma que:

- **Message ID** - identificador de mensagem (número inteiro entre 1 e 65535, gerado aleatoriamente pelo CL ou servidor que faz a query original) que irá ser usado para relacionar as respostas recebidas com a query original;
- **FLAGS** - devem ser suportadas as flags Q, R e A; a flag Q ativa indica que a mensagem é uma query, senão é uma resposta a uma query; se a flag R estiver ativa na query indica que se deseja que o processo opere de forma recursiva e não iterativa (que é o modo por defeito); se a flag R estiver ativa na resposta indica que o servidor que respondeu suporta o modo recursivo; se a flag A estiver ativa na resposta indica que a resposta é autoritativa (o valor da flag A é ignorada nas queries originais);
- **RESPONSE CODE** - indica o código de erro na resposta a uma query; se o valor for zero então não existe qualquer tipo de erro e a resposta contém informação que responde diretamente à query; a resposta deve ser guardada em cache; se houver erros, o sistema deve suportar os seguintes códigos de erro :
 - erro 1, o domínio incluído em NAME existe mas não foi encontrada qualquer informação direta com um tipo de valor igual a TYPE OF VALUE (o campo de resultados vem vazio mas o campo com a lista de valores de autoridades válidas para o domínio e o campo com a lista de valores com informação extra podem ser incluídos); este caso é identificado como resposta negativa e pode ser guardada em cache para que a um pedido semelhante, num horizonte temporal curto, o servidor possa responder mais rapidamente;
 - erro 2, o domínio incluído em NAME não existe (o campo de resultados vem vazio mas o campo com a lista de valores de autoridades válidas onde a resposta foi obtida e o campo com a lista de valores com informação extra podem ser incluídos); este caso também é identificado como resposta negativa e pode ser guardada em cache;
 - erro 3, a mensagem DNS não foi decodificada corretamente;

- **NUMBER OF VALUES** - número de entradas relevantes (num máximo de 255) que respondem diretamente à query (i.e., as entradas a cache ou na base de dados do servidor autoritativo e que têm um parâmetro igual a NAME e um tipo de valor igual a TYPE OF VALUE) e que fazem parte da lista de entradas incluídas no campo RESPONSE VALUES;
- **NUMBER OF AUTHORITIES** - número de entradas (num máximo de 255) que identificam os servidores autoritativos para o domínio incluído no RESULT VALUES;
- **NUMBER OF EXTRA VALUES** - número de entradas (num máximo de 255) com informação adicional relacionada com os resultados da query ou com os servidores da lista de autoridades;
- **QUERY.INFO** - informação do parâmetro da query (NAME) e o tipo de valor associado ao parâmetro (TYPE OF VALUE); os tipos suportados são os mesmos suportados na sintaxe dos ficheiros de base de dados dos SP; na resposta a queries, os servidores devem copiar a informação do QUERY INFO e incluí-la na mensagem de resposta;
- **RESPONSE VALUES** - lista das entradas que fazem match no NAME e TYPE OF VALUE incluídos na cache ou na base de dados do servidor autoritativo; cada entrada deve ter a informação completa tal como é definida na base de dados DNS do SP do domínio referente ao NAME;
- **AUTHORITIES VALUES** - lista das entradas que fazem match com o NAME e com o tipo de valor igual a NS incluídos na cache ou na base de dados do servidor autoritativo; cada entrada deve ter a informação completa tal como é definida na base de dados DNS do SP do domínio referente ao NAME;
- **EXTRA VALUES** - lista das entradas do tipo A (incluídos na cache ou na base de dados do servidor autoritativo) e que fazem match no parâmetro com todos os valores no campo RESPONSE VALUES e no campo AUTHORITIES VALUES de forma a que o elemento que o CL ou servidor que recebe a resposta não tenha que fazer novas queries para saber os endereços IP dos parâmetros que vêm como valores nos outros dois campos; cada entrada deve ter a informação completa tal como é definida na base de dados DNS do SP do domínio referente ao NAME.

3.3 Tratamento de erros

Quanto ao tratamento de erros o nosso sistema permite o tratamentos de varios erros possiveis de ocorrer, dos quais :

- **Erros no Cliente** A mensagem do cliente é analisada para verificar se a estrutura da mensagem enviada esta bem contruida segundo a estrutura de uma mensagem DNS, incluindo o numero correto de elementos e argumentos válidos. Em caso de erro a mensagem será eliminada e o cliente encerado.

- **Erros nas queries** Caso o domínio onde é realizada a query é o correto segundo o NAME da query mas não se encontre qualquer tipo de valor relacionado ao TYPE OF VALUE a query ativa a flag de erro 1. Por outro lado, no caso domínio procurado não existir é retornado o erro 2. Outro erro possível é o erro de decodificação que é marcado com a flag a 3. Por fim, se não existir nenhum destes erros a flag mantém-se a 0.

Todos os erros encontrados serão registados no ficheiro de logs, sendo o ficheiro (all.log) regista as queries que não pertencem ao domínio e o segundo o ficheiro log do componente onde o erro ocorreu, com exceção de erros no cliente.

3.4 Mecanismo de Codificação Binária

O mecanismo de codificação binária implementado tem como objetivo reduzir o tamanho das mensagens de query enviadas. O problema pode ser dividido em 2 dois métodos o `query.encode()`, que codifica uma query em bytes, e o `query.decode(data)`., que decodifica uma sequência de bytes num objeto query. De modo a executar uma codificação bem sucedida há que identificar as secções da query que podem ser comprimidas.

Como é apresentado no ponto 3.2, uma mensagem de DNS tem dois campos gerais o *HEADER* e a *DATA*.

O campo *HEADER* tem os seguintes campos:

- Message ID - números de 0 a 65535 (16 bits)
- Flags - Vazio/ "Q" / "Q+R" / "A" / "A+R" (3 bits)
- Response Code : 0 / 1 / 2 / 3 (2 bits)
- Number of Response Values: 0-255 (8 bits)
- Number of Authorities Values: 0-255 (8 bits)
- Number of Extra Values: 0-255 (8 bits)

A codificação binária personalizada utiliza o tamanho máximo de cada campo e junta tudo num *bitarray*. Deste modo não se utiliza os separadores e o tamanho de 1 byte por carácter, e assim, consegue-se diminuir de forma significativa o tamanho da string.

Veja-se o caso da query de exemplo e tamanho que ocupa sem/com codificação binária personalizada:

65535,Q+R,0,0,0,0;example.com.,MX;

Sem codificação binária personalizada: 45 bytes

Com codificação binária personalizada : 18 bytes

Como é possível observar pela query exemplo, com codificação binária personalizada houve uma melhoria de eficiencia de 250 %

Como o campo *DATA* é variável não é possível definir barreiras concretas, por isso, a parte do *DATA* é sempre codificada em UTF-8. Depois concatena-se as duas partes e tem-se a query de dns totalmente codificada.

4 Modelo de Comunicação

4.1 Protocolos de Transporte

A ligação entre componentes é estabelecida através de dois protocolos de transporte diferentes, o protocolo UDP e o protocolo TCP.

- **Protocolo UDP** : utilizado nas `QUERIES DE DNS`. A maior parte da comunicação entre os componentes do sistema utiliza o protocolo não orientado à conexão. O UDP permite que a aplicação envie um datagrama encapsulado num pacote IPv4 para um destino muito rapidamente, porém sem qualquer tipo de garantia que o pacote chega corretamente.
- **Protocolo TCP** : utilizado na `TRANSFERÊNCIA DE ZONA`. O protocolo TCP é um protocolo orientado à conexão e, portanto, inclui vários mecanismos para iniciar, manter e encerrar a comunicação, negociar tamanhos de pacotes, detectar e corrigir erros, evitar congestionamento do fluxo e permitir a retransmissão de pacotes corrompidos, independente da qualidade do meio físicos. A transferência de zona trata-se de um mecanismo de atualização da base de dados do Servidor Secundário. Quando o SS verifica que a sua BD está desatualizada envia um pedido de uma cópia da BD do SP do seu domínio. É importante assegurar a conexão, daí utilizar-se o TCP, para garantir que o SS não tem informação corrompida.

4.2 Processamento Iterativo

O DNS iterativo é um tipo de serviço DNS que permite que um servidor DNS resolva uma consulta DNS solicitando iterativamente uma série de servidores DNS pela resposta, começando com os servidores de Topo e descendo até os servidores de nomes autorizados para o domínio perguntado.

Quando um cliente envia uma solicitação de um site para um servidor DNS iterativo, o servidor de resolução primeiro verifica a sua cache para ver se possui uma cópia do registro DNS do site solicitado. Caso contrário, o servidor enviará uma solicitação aos servidores DNS raiz (ST) para localizar o endereço dos servidores de domínio de topo (SDT). O servidor iterativo enviará uma solicitação ao SDT, que podem fornecer o endereço dos servidores de nomes autorizados ou podem fornecer uma referência a outro servidor mais próximo dos servidores de nomes autorizados. O processo continua até que o servidor iterativo receba uma resposta dos servidores de nomes autorizados, que fornecerão o endereço IP do site. O servidor de resolução

retornará esse endereço IP ao cliente, concluindo o processo de resolução de DNS.

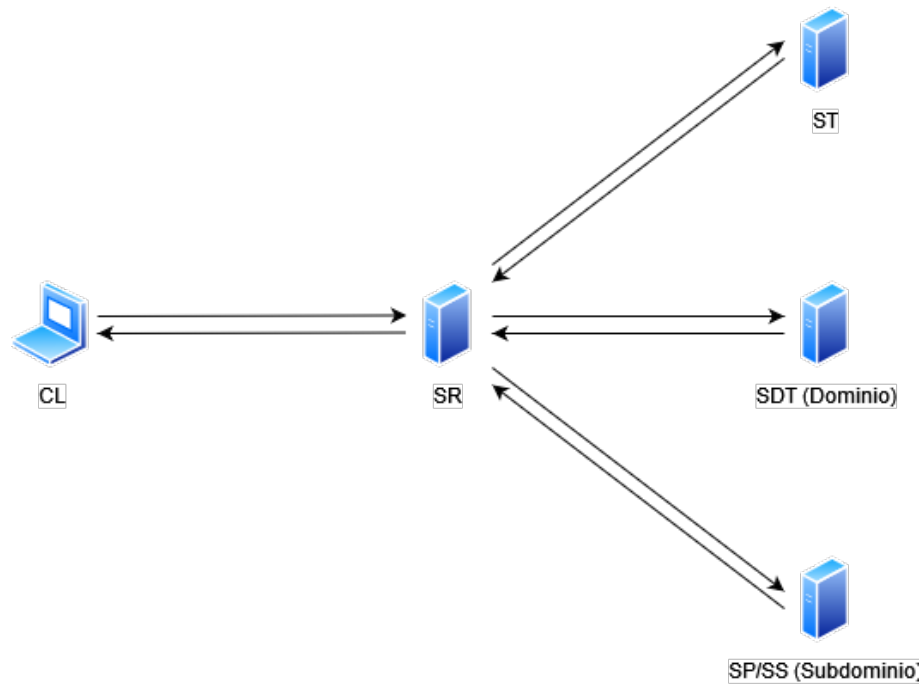


Figura 4.1: Processamento Iterativo

4.3 Processamento Recursivo

O DNS recursivo é um tipo de serviço DNS que permite que um servidor DNS resolva uma consulta DNS pesquisando recursivamente na hierarquia DNS para encontrar a resposta. Quando um cliente envia uma solicitação de um site para um servidor DNS recursivo, o servidor primeiro verifica seu cache para ver se possui uma cópia do registro DNS do site solicitado. Caso contrário, o servidor enviará uma solicitação aos servidores de topo (ST) para localizar o endereço dos servidores de domínio de topo (SDT) para o domínio em questão. O ST enviará uma solicitação aos SDT, que fornecerão o endereço dos servidores de nomes autorizados para o domínio. O SDT enviará uma solicitação aos servidores de nomes autorizados (SP), que fornecerão o endereço IP do site. O servidor recursivo irá então devolver este endereço IP ao cliente, completando o processo de resolução de DNS.

Observando a imagem 4.2 consegue-se perceber melhor o processo.

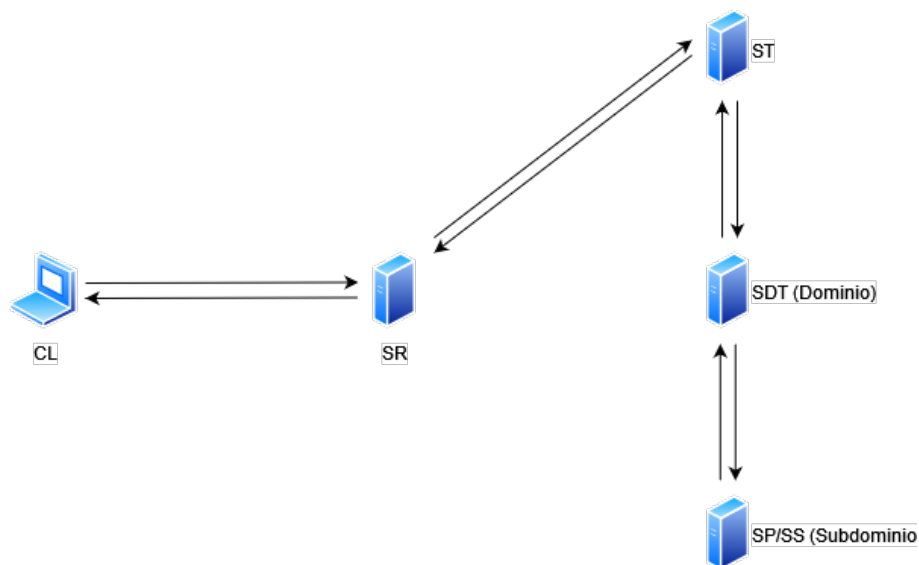


Figura 4.2: Processamento Recursivo

4.4 Processamento das Queries

Todas as interações começam com o envio duma query DNS para um servidor. Essa query vem dum CL ou dum outro qualquer servidor DNS. A query é codificada no envio e decodificada na receção. Na mensagem da query DNS só são usados os campos assinalados com '*'. Os restantes campos do cabeçalho são ignorados (campos devem ser colocados a zero) e os restantes campos dos dados são nulos (não são sequer incluídos) na mensagem.

Um servidor deve processar a query recebida e se a decodificação da informação da query for correta o servidor deve tentar encontrar na sua cache a informação direta que responda à query.

4.5 Ambiente de Teste

O Ambiente de Teste é composto pelos ficheiros de configuração e pelos ficheiros de Base de Dados de todos os componentes apresentados na topologia. Como mencionado anteriormente, o ficheiro de configuração estipula o comportamento do componente de software e as suas base de dados a sua capacidade de responder a certas queries. O grupo construi:

- **Ficheiros de Configuração** - 15 Ficheiros: 4 Servidores Primários, 8 Servidores Secundários, 2 Servidores de Topo e 1 Servidor de DNS Reverso.
 - ST1.conf - Servidor de Topo 1 (.)
 - ST2.conf - Servidor de Topo 2 (.)
 - SP-bra.conf - Servidor Primário Braga (bra.)

- SP-brg.conf - Servidor Primário Bragança (brg.)
 - SP-bcl.conf - Servidor Primário Barcelos (bcl.)
 - SP-mdd.conf - Servidor Primário Miranda do Douro (mdd.)
 - SS1-bra.conf - Servidor Secundário Braga (bra.)
 - SS2-bra.conf - Servidor Secundário Braga (bra.)
 - SS1-brg.conf - Servidor Secundário Bragança (brg.)
 - SS2-brg.conf - Servidor Secundário Bragança (brg.)
 - SS1-bcl.conf - Servidor Secundário Barcelos (bcl.)
 - SS2-bcl.conf - Servidor Secundário Barcelos (bcl.)
 - SS1-mdd.conf - Servidor Secundário Miranda do Douro (mdd.)
 - SS2-mdd.conf - Servidor Secundário Miranda do Douro (mdd.)
 - SR-mdd.conf - Servidor de Resolução
- **Ficheiros de Base de Dados** - 6 Ficheiros : 4 Servidores Primários, 2 para Servidores de Topo e 1 Lista de servidores de topo.
 - mdd.db - Servidor Primário Braga (mdd.)
 - bcl.db - Servidor Primário Bragança (bcl.)
 - bra.db - Servidor Primário Braga (bra.)
 - brg.db - Servidor Primário Bragança (brg.)
 - st1.db - Servidor de Topo 1 (.)
 - st2.db - Servidor de Topo 2 (.)
 - rootservers.db - Lista de servidores de topo

4.5.1 Ficheiros de Configuração

Os ficheiros de configuração foram todos contruídos com a formatação sintática apresentada no capítulo "Modelo de Informação" na subsecção "Ficheiros de Configuração SP, SS e SR".

O exemplo apresentado corresponde ao Ficheiro de Configuração do Servidor Principal do domínio .bra

```
# Configuration file for primary server for bra.
bra.  DB ./db_files/bra.db
bra.  SS 10.0.17.12
bra.  SS 10.0.16.11
bra.  DD 127.0.0.1
bra.  LG ./log_files/bra.log
all LG ./log_files/all.log
root ST ./db_files/rootservers.db
```

Example 1: Ficheiro de Configuração do SP .bra

Como se pode ver no exemplo o SP do domínio .bra tem na sua configuração os endereços dos SS com quem partilha o domínio e o endereço de domínio default, assim como, tem os caminhos para os ficheiros relevantes: a lista de ST, a sua Base de Dados, o ficheiro de Logs do componente e o de Logs geral.

Depois de ser analisado, todas as informações são guardadas em memória volátil numa estrutura de dados `config()`.

4.5.2 Ficheiros de Base de Dados

Os ficheiros de configuração foram todos contruídos com a formatação sintática apresentada no capítulo "Modelo de Informação" na subsecção "Ficheiros de Dados do Servidor Primário". O exemplo apresentado corresponde ao ficheiro de Base de Dados do SP do domínio .bra.

```

# DNS database file for domain .bra
# It also includes a pointer to the primary server
# of the bcl.bra subdomain

@ DEFAULT bra.
TTL DEFAULT 86400

@ SOASP ns1.bra TTL
@ SOADMIN dns\.admin.bra. TTL
@ SOASERIAL 0117102022 TTL
@ SOAREFRESH 14400 TTL
@ SOARETRY 3600 TTL
@ SOAEXPIRE 604800 TTL
@ NS ns1.bra. TTL
@ NS ns2.bra. TTL
@ NS ns3.bra. TTL

bcl.bra. NS sp.bcl.bra.

@ MX mx1.bra. TTL 10
@ MX mx2.bra. TTL 20

ns1 A 10.0.12.10 TTL
ns2 A 10.0.17.12 TTL
ns3 A 10.0.16.11 TTL
sp A 10.0.10.10 TTL
mx1 A 10.0.19.10 TTL
mx2 A 10.0.16.12 TTL
www A 10.0.18.11 TTL 200
ftp A 193.136.130.20 TTL

sp CNAME ns1 TTL
ss1 CNAME ns2 TTL
ss2 CNAME ns3 TTL
mail1 CNAME mx1 TTL
mail2 CNAME mx2 TTL

```

Example 2: Ficheiro de Base de Dados do SP .bra

5 Funcionamento dos Componentes

5.1 Execução Concorrente de Pedidos

De modo a cumprir a execução concorrente de pedidos no servidor foi necessário implementar multithreading em todos os servidores, sendo que o grupo escolheu a abordagem 1-thread-por-pedido.

A estratégia tomada tem em conta um socket UDP de receção e envio de pedidos. Sempre que se recebe uma nova mensagem identifica-se se é um "request" ou uma "response", e são inseridos na requestpool e na responsepool, respetivamente. Os requests são executado em threads de processamento de pedidos enquanto o response ativa um evento que faz as threads bloqueadas verificar o responsepool (para ver se uma resposta da sua thread chegou). Os pedidos de transferência de zona são abordados como partes do processamento de uma query. Deste modo mantém-se um sistema que consegue receber/processar/enviar pedidos concorrentemente.

5.1.1 Estado Partilhado e Zona de Exclusão Mútua

De modo a manter o estado partilhado seguro e sem possíveis condições de corrida ou deadlocks implementou-se vários tipos de lock:

- **lockUDP**: um ReentrantLock que é adquirido sempre que se pretende escrever no socket.
- **lockTCP**: um ReentrantLock que é adquirido nos momentos de transferência de zona.
- **lock do Logger**: um ReentrantLock que é adquirido nos momentos de escrita nos ficheiros de logs.
- **ReadWriteLock()** : um ReadWriteLock que aproveita a assimetria de situações de leitura e de escrita. Utilizado nas seguintes estruturas:
 - Cache : para ler, registar e eliminar cacheEntries.
 - POOL de Pedidos : para ler, inserir e remover pedidos.

5.2 Processamento de uma querie

5.3 Cliente

O cliente constrói queries de DNS e envia-as para o endereço IP indicado como *input*. De seguida, fica à espera de uma resposta na forma de uma querie de DNS. Em caso de erro de formatação no input ou na resposta recebida, o cliente encerra o processo com uma mensagem de erros.

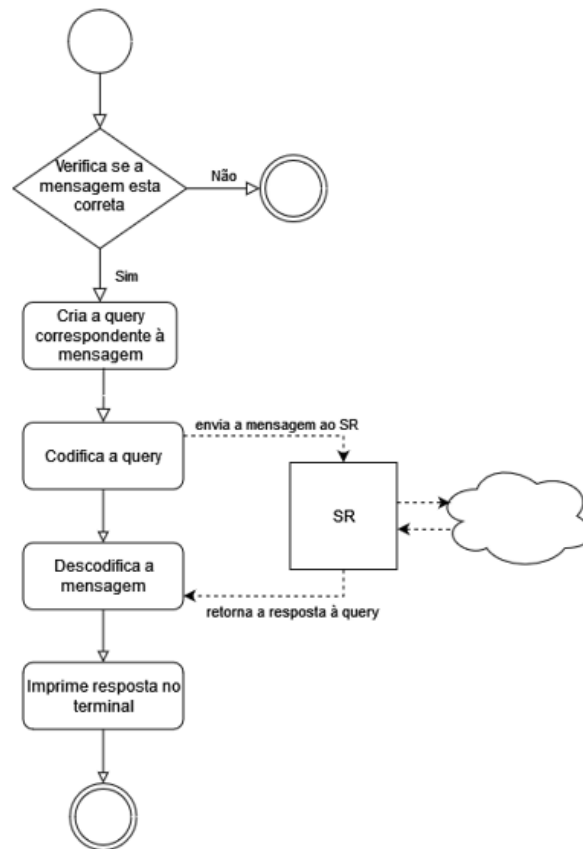


Figura 5.1: Diagrama de Atividades do CL

5.4 Servidor Resolução (SR)

O servidor de resolução recebe mensagens de clientes/servidores/servidores de topo, separa-as em duas *pool*, uma para as respostas e outra para os pedidos. Se receber uma resposta sinaliza as threads bloqueadas que existe uma nova resposta. Por outro lado, se receber um pedido inicia uma thread para poder executar os pedidos paralelamente. Esta thread verifica se a query é recursiva ou iterativa e inicia o respetivo processo.

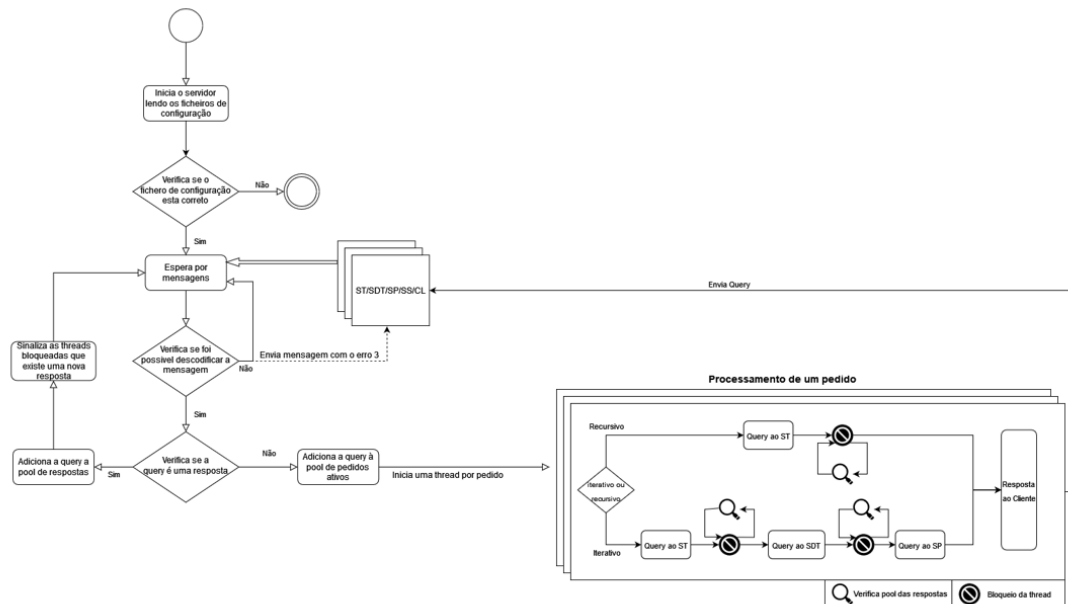


Figura 5.2: Diagrama de Atividades do SR

5.5 Servidor Secundário (SS)

O SS utiliza os dois protocolos de comunicação. Nas suas queries de DNS normais utiliza o UDP, quando recebe ou envia queries sobre o seu domínio. Na transferência de zona utiliza o protocolo TCP.

O funcionamento simplificado do SS passa por depois de executar os procedimentos padrão (ler ficheiro de configuração, alocar espaço para a cache, registar nos logs). No arranque o SS faz um pedido ao SP para o envio da sua Base de Dados. A transferência de zona do lado do SS consiste no envio de uma mensagem a indicar o domínio no qual está interessado em copiar a BD, depois de verificado, recebe o número de entradas a receber, confirma o número e por último recebe linha a linha o ficheiro de Base de Dados do SP do domínio.

Depois da operação estar encerrada, encontra-se preparado para receber e responder a queries de DNS, até à próxima verificação denotar que a sua BD está desatualizada. O funcionamento do SS é possível observar no diagrama da figura 5.3 .

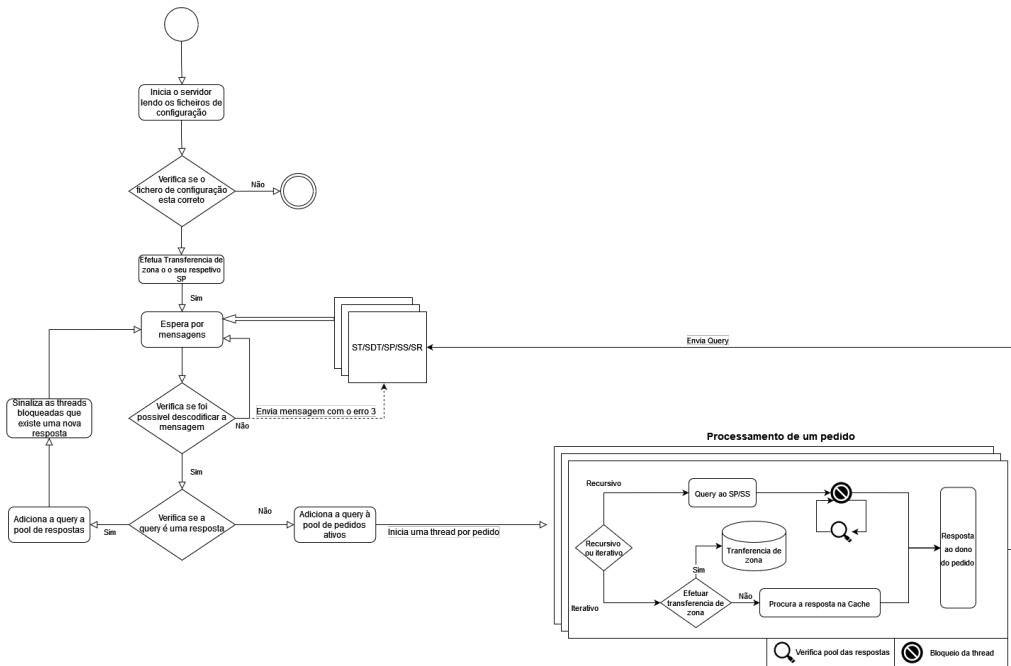


Figura 5.3: Diagrama de Atividades do SS

5.6 Servidor Primário (SP)

O SP também utiliza os dois protocolos de comunicação. Nas suas queries de DNS normais utiliza o UDP, quando recebe ou envia queries sobre o seu domínio. Na transferência de zona utiliza o protocolo TCP.

O funcionamento simplificado do SP passa por receber e responder a queries, no entanto, assim que recebe uma query "DBU" (DataBaseUpdate request) o SP inicia o processo de transferência de zona. A transferência de zona do lado do SP consiste na receção de uma mensagem a indicar o domínio que o cliente pretende copiar a DB, depois verifica que o seu domínio e o indicado são iguais, e envia o número de entradas a transferir, recebe a confirmação que o SS está pronto e por último envia linha a linha do seu ficheiro de Base de Dados.

Depois da operação estar encerrada, continua preparado para receber e responder a queries de DNS. O SP atua como ST/SDT/SP dependendo do contexto inserido. Semelhante ao funcionamento geral do SS, o SP é possível observar no diagrama da figura 5.3 .

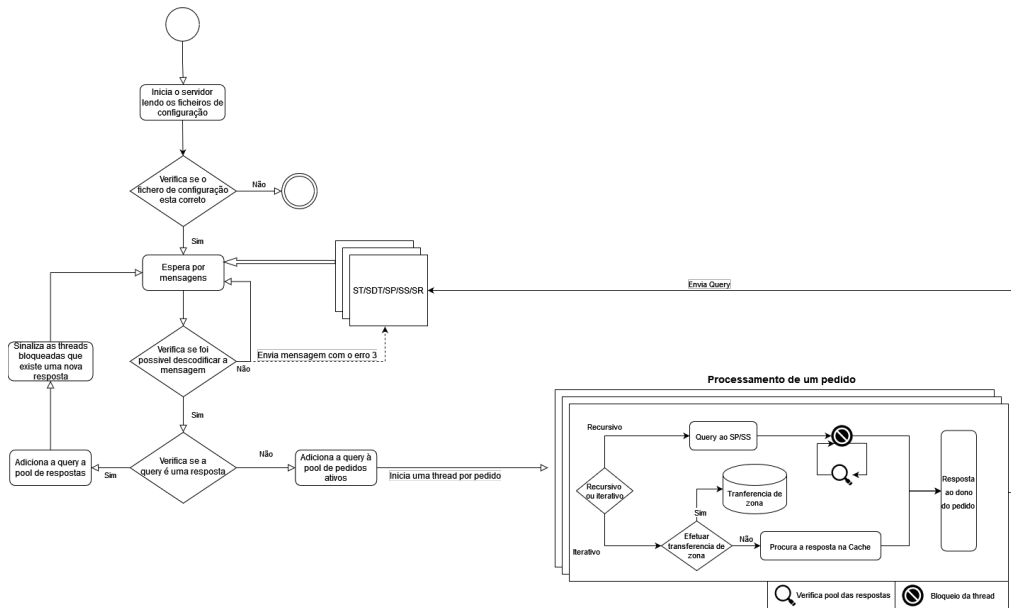


Figura 5.4: Diagrama de Atividades do SP

5.7 Servidor de DNS Reverso

Para executar uma pesquisa DNS reversa, um cliente envia uma query a um servidor DNS, solicitando que ele resolva um endereço IP para um nome de domínio. O servidor DNS então realiza uma pesquisa no seu banco de dados para localizar quaisquer registos PTR (ponteiro) que correspondam ao endereço IP da query. Se for encontrada uma correspondência, o servidor DNS retorna o nome de domínio correspondente ao cliente. Se nenhuma correspondência for encontrada, o servidor DNS retornará uma mensagem "registro não encontrado".

6 Manual de Utilização

Para verificar as funcionalidades do sistema basta executar o seguinte código.

```
$ python3 ../../Projeto-CC/client.py 10.0.21.13 bcl.bra. MX
```

Example 3: Execução do Cliente

```
$ python3 ../../Projeto-CC/server.py ../config_files/SR.conf 7000  
{timetout} {DEBUG}
```

Example 4: Execução do SR-mdd

```
$ python3 ../../Projeto-CC/server.py ../config_files/SP-bcl.conf  
9000 {timetout} {DEBUG}
```

Example 5: Execução do SP-bcl

```
$ python3 ../../Projeto-CC/server.py ../config_files/SS1-bcl.conf  
9000 {timetout} {DEBUG}
```

Example 6: Execução do SS1-bcl

```
$ python3 ../../Projeto-CC/server.py ../config_files/SP-bra.conf  
9000 {timetout} {DEBUG}
```

Example 7: Execução do SP-bra

```
$ python3 ../../Projeto-CC/server.py ../config_files/ST1.conf  
10000 {timetout} {DEBUG}
```

Example 8: Execução do ST1

Observação: O código enviado encontra-se com formato para ser testado na própria máquina e não no core. Para se testar no core deve-se trocar algumas secções do comentário.

7 Conclusão

Em conclusão, o trabalho feito para construir um sistema DNS foi extenso e multifacetado.

O projeto começou com uma análise completa do estado atual do DNS e sua arquitetura subjacente. Isso foi seguido pelo desenvolvimento de um conjunto de requisitos e objetivos de design para o novo sistema DNS, que foram usados para orientar o processo de implementação.

Face ao calendário altamente preenchido e com elevada carga de trabalho o grupo fez tudo o que estava ao seu alcance para apresentar o seu melhor trabalho. O grupo também é da opinião que o DNS apesar de ser um excelente tema para nos debruçarmos num trabalho prático, o nível de pormenor e detalhe do enunciado, por vezes sem um objetivo prático em vista, não foi o adequado.

8 Anexos

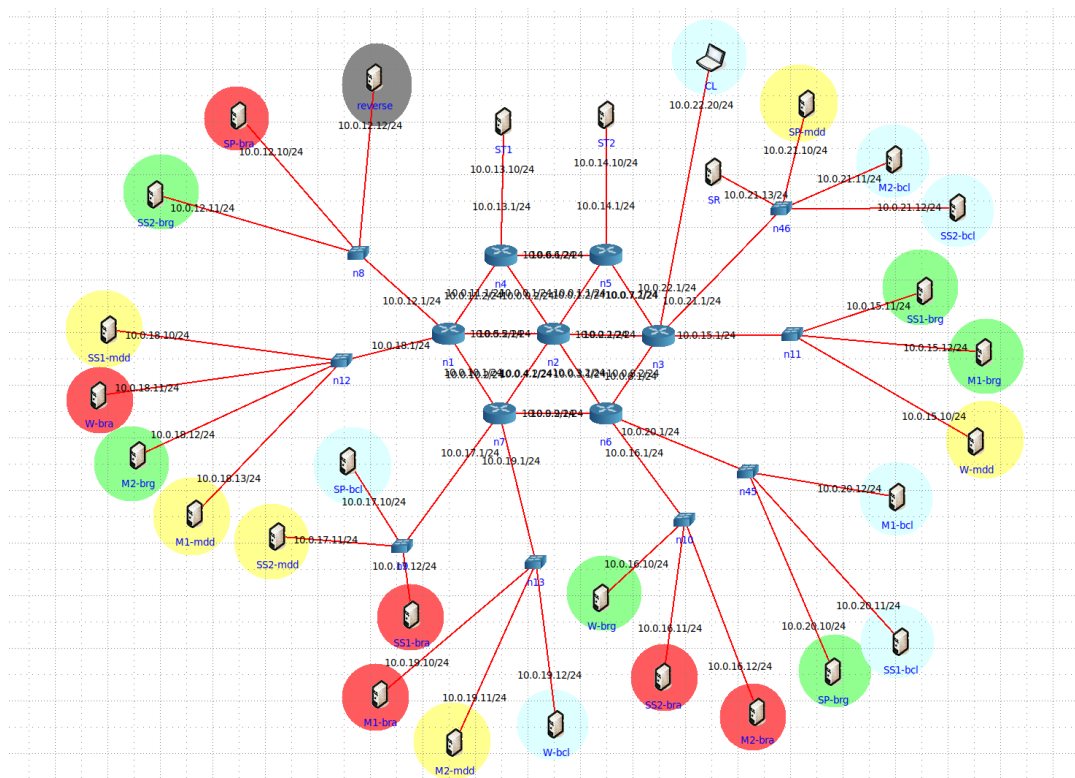


Figura 8.1: Topologia da rede concebida

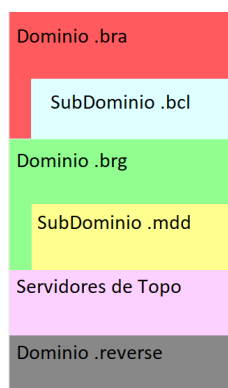


Figura 8.2: Legenda da rede concebida

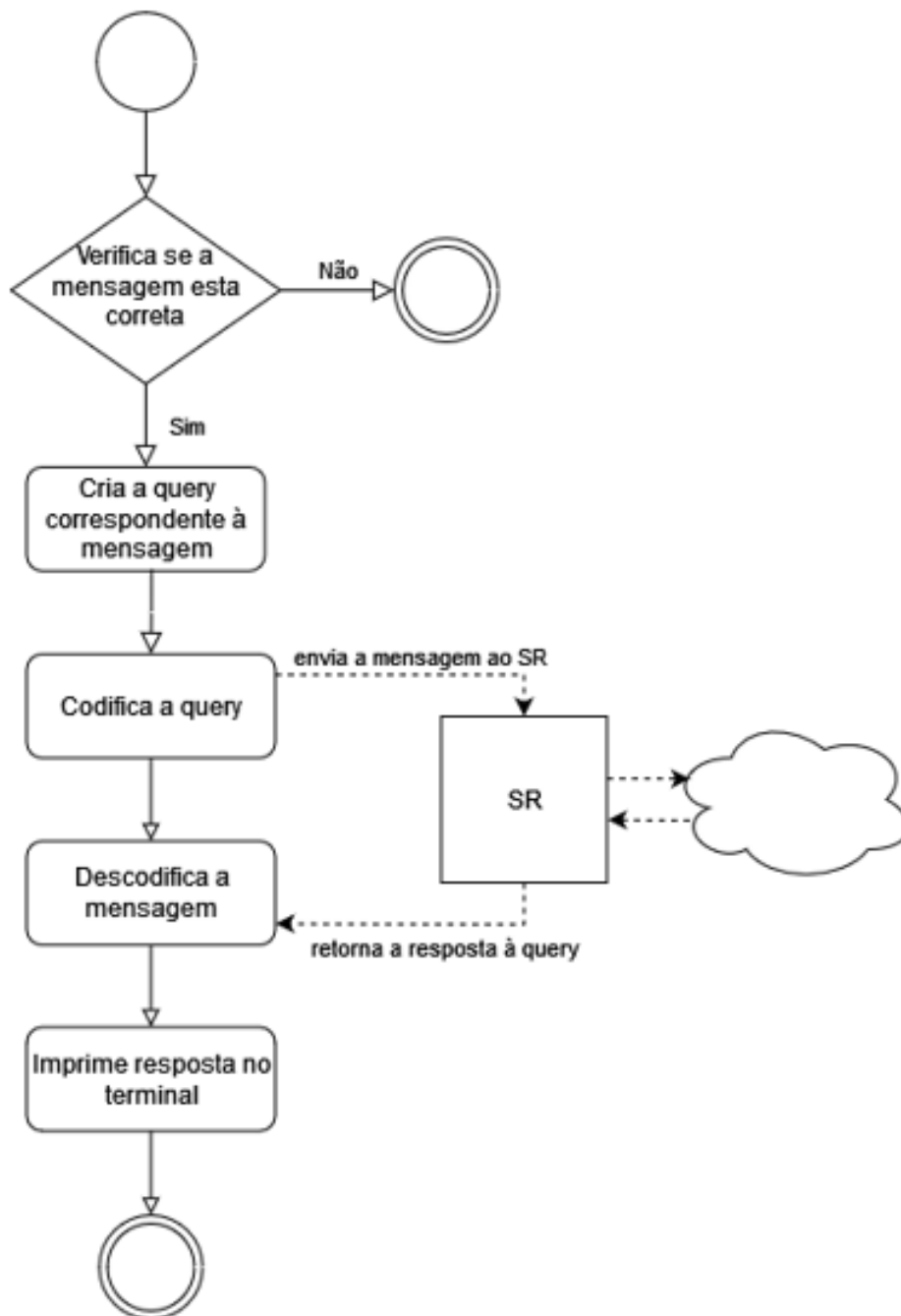


Figura 8.3: Diagrama do Cliente

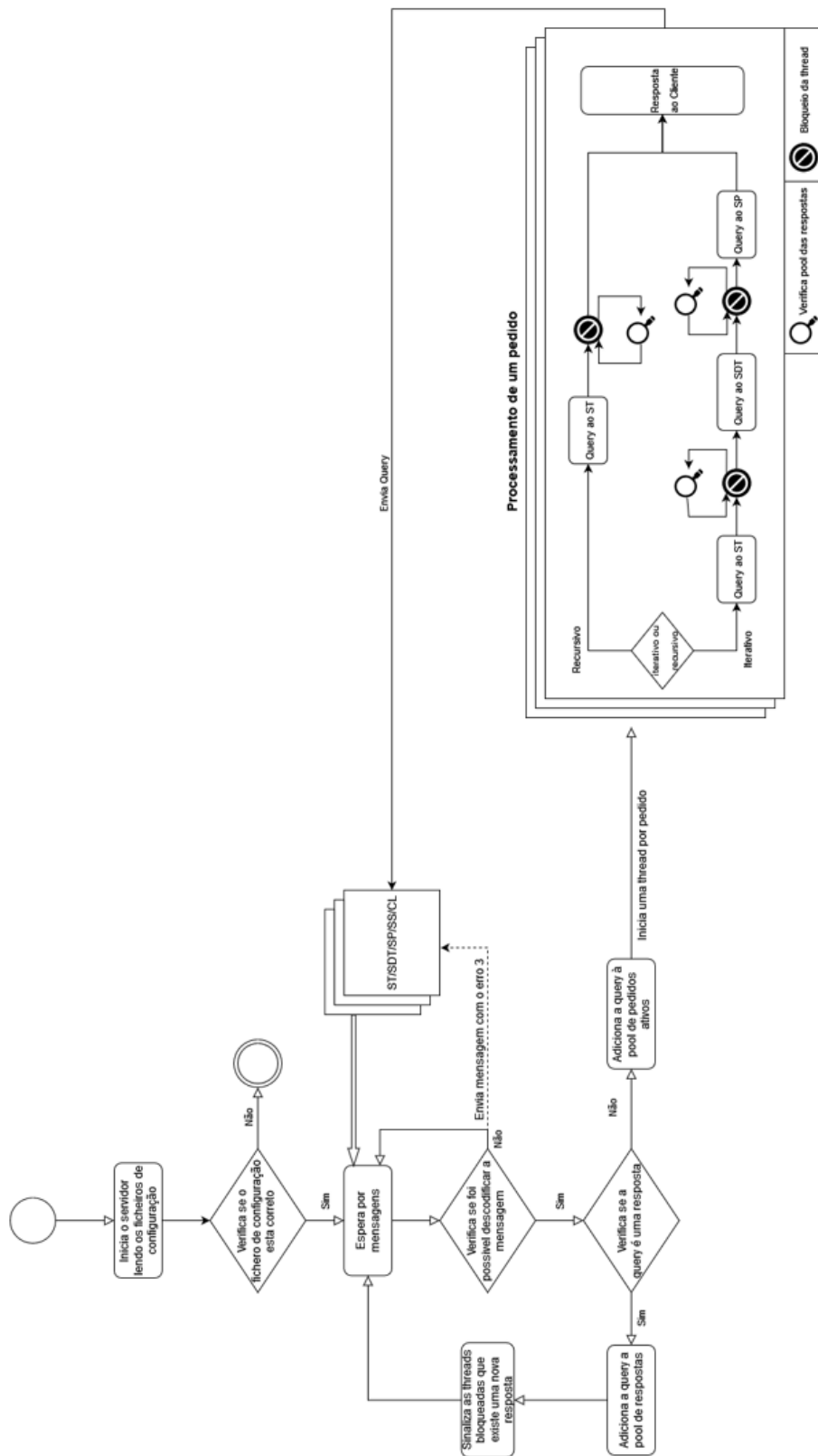


Figura 8.4: Diagrama do Servidor Resolução

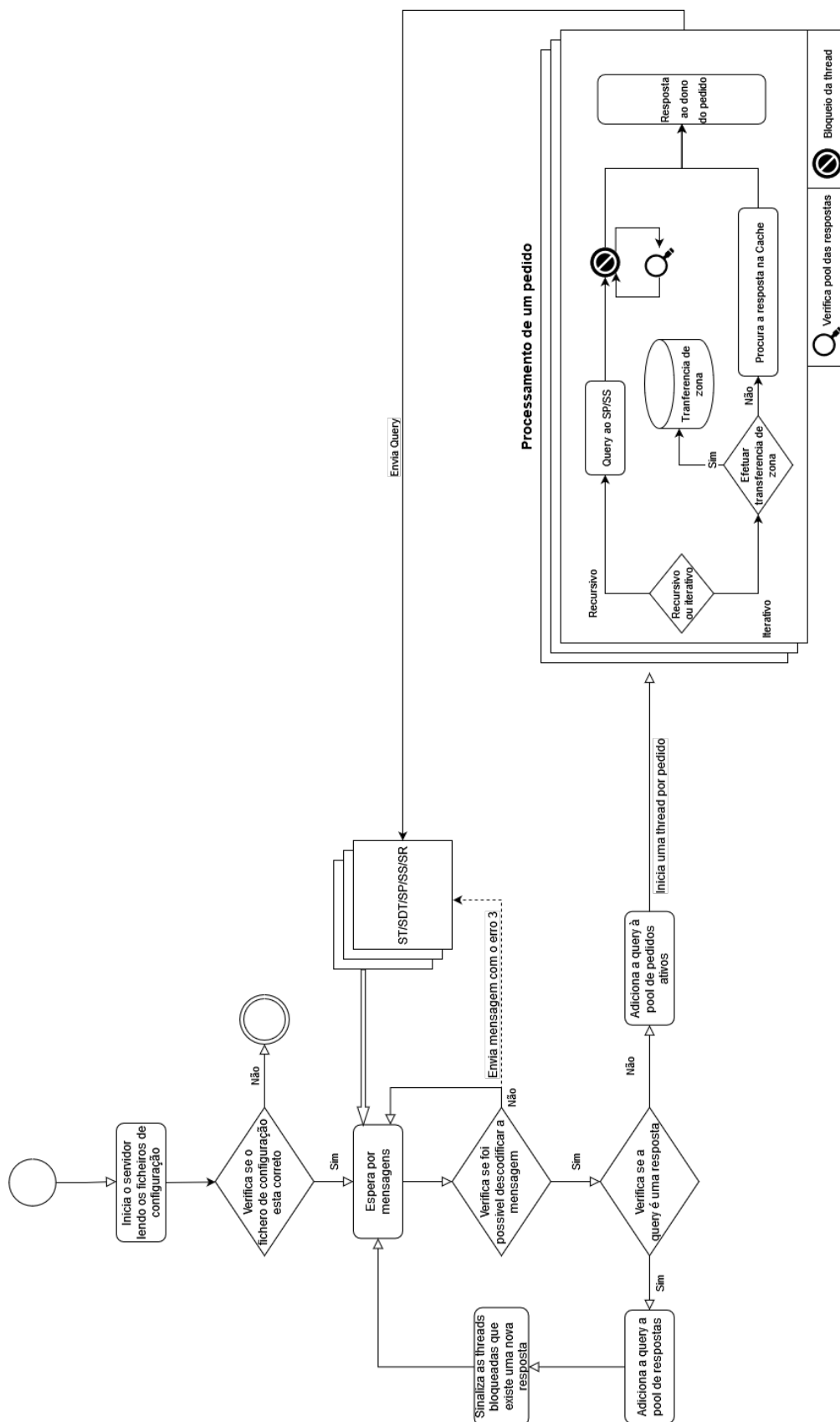


Figura 8.5: Diagrama do Servidor Primário

